International Journal of Operations Research

International Journal of Operations Research Vol. 7, No. 2, 45–60 (2010)

A Search Algorithm for Solving the Joint Replenishment Problem in a Distribution Center with Warehouse-space Restrictions

Ming-Jong Yao*

Department of Transportation Technology and Management, National Chiao Tung University,

1001 University Road, 30010 Hsinchu, Taiwan

Received September 2010; Revised October 2010; Accepted November 2010

Abstract—This study is an extension of the Joint Replenishment Problem (JRP) that takes into accounts the warehouse-space restrictions in a distribution center. The focus of this study is to determine the lot sizes of each product under power-of-two policy to minimize the total costs per unit time and to generate a feasible replenishment schedule of multiple products without exceeding the available warehouse-space of the distribution center. To solve this problem, we propose a search algorithm to obtain the optimal replenishment cycle times of multiple products. Also, we devise a new heuristic to generate feasible replenishment schedules for the solutions obtained by the search algorithm. Based on our numerical experiments, we conclude that the proposed search algorithm could effectively solve the JRP with warehouse-space restrictions.

Keywords—Inventory, lot sizes, scheduling, search algorithm, warehouse, restrictions

1. INTRODUCTION

In this paper, we are interested in obtaining an optimal replenishment strategy for a distribution center with limited warehouse space available for inventory storage. The decision maker concerns with the determination of lot sizes and replenishment schedules of n products in the distribution center. The focus of this study is to optimally coordinate the replenishment of products so as to minimize the total costs incurred per unit time without violating the warehouse-space restrictions.

This study is actually an extension of a well-known lot sizing and scheduling problem, *viz.*, the Joint Replenishment Problem (JRP). In the decision-making scenario of this study, we drop off an impractical assumption in the conventional JRP, namely, the distribution center has infinite warehouse space available. Without considering the warehouse-space restrictions, one could encounter an infeasible replenishment strategy for the practice in the distribution center even he/she is able to obtain an optimal solution of the conventional JRP. Therefore, we are motivated to study the JRP considering the warehouse-space restrictions and to propose a solution approach to support the managers' decision making in the distribution center.

The rest of the paper is organized as follows. Section 2 reviews the literature relative to the JRP with the warehouse-space restrictions. Then, we discuss the derivation of the mathematical model for the JRP with the warehouse-space restrictions in Section 3. Section 4 presents the proposed solution approaches for solving the problem. Next, in the first part of Section 5, we present a numerical example to demonstrate the implementation of the proposed search algorithm. Also, based on our random experiments, we show that the proposed search algorithm could effectively obtain solutions with excellent quality in the second part of Section 5. Finally, we address our concluding remarks in Section 6.

2. LITERATURE REVIEW

In this section, we review the problem definition of the JRP and examine the solution approaches for the JRP. Also, we survey the lot scheduling problems with the warehouse-space restrictions in the literature.

* Corresponding author's email: myaoie@gmail.com

IJOR Vol. 7, No. 2, 45−60 (2010)

2.1 The problem definition and the solution approaches for the JRP

The Joint Replenishment Problem (JRP) is concerned with the determination of lot sizes and schedule of n products in single-facility production/distribution systems over an infinite (and continuous) planning horizon. The objective of the JRP is to minimize the total costs incurred per unit time. The costs considered generally include *setup costs* and *inventory holding costs*.

For each product i, its demand rate d_i is fixed (and continuous), and each unit incurs a holding cost h_i per unit time. Since usually it takes an extremely short time for unloading trucks in the distribution center, it is assumed that the replenishment for each product is *instantaneous*. Also, two types of setup costs are considered in the JRP: (*i*) a *major setup cost*, denoted by A, is incurred whenever the distribution center sets up to jointly replenish a subset of products; and (*ii*) a *minor setup cost* a_i is incurred while each product i is replenished.

For decision makers facing the JRP, an intuitive move is to jointly replenish many products in each major setup to share the major setup cost so as to minimize the average total costs. Therefore, the key concern of the JRP is to coordinate the replenishment schedule of each product i to economically share major setup cost, and balance the holding costs from the inventory of jointly replenished products.

In the literature, researchers have been devoting their efforts to the studies of the JRP for decades. One may refer to Goyal and Satir (1989) as well as Aksoy and Erenguc (1988) for the details on the problem definition and the early studies of the JRP. The solution approaches for the JRP usually assume that each product i is replenished after a fixed cycle, denoted by T_i where T_i is the length of time between two consecutive minor setups for product i. Most of early studies assume that T_i is equal to a positive integer k_i times B, *i.e.*, $T_i = k_i B$, where B is a basic period in the planning horizon. Also, it is usually assumed that the replenishment frequency for major setup, denoted by k_0, is always set to 1 in the JRP. In order to obtain an optimal solution, one must simultaneously determine a value of B and a vector of optimal (integer) multipliers $(k_1, k_2, \ldots, k_n)$ in the JRP.

Based on the assumptions discussed above, the mathematical model for the JRP is formulated as follows.

$$\textit{Minimize}\quad TC_{GI}(k_1, k_2, \ldots, k_n, B) = \frac{1}{B}\left(\frac{A}{k_0} + \sum_{i=1}^{n}\frac{a_i}{k_i}\right) + \frac{B}{2}\sum_{i=1}^{n} k_i d_i h_i \qquad (a) \tag{1}$$

$$\text{subject to} \qquad k_0 = 1; k_i \in \mathbb{N}^+, \text{for } i = 1, \ldots, n. \qquad (b)$$

The subscript GI in the objective function $TC_{GI}(k_1, k_2, ..., k_n, B)$ indicates that the JRP model is formulated under *General Integer* (*GI*) *policy*, which is expressed by constraint (*1b*). *GI* policy requires that all the k_i's must be positive integers. Therefore, the JRP model in (1) is a nonlinear, integer program.

The JRP has been studied for some thirty years. Extensive research efforts have been addressed to attempt efficient heuristics for solving the JRP. Goyal (1973, 1974) proposed an enumeration approach, and he claimed that it always obtains a global optimal solution (though without proof). The essence of his enumeration approach is as follows. Enumerate B and k_i for each product i so as to satisfy both of the following conditions:

$$TC_{GI}(k_i(B), B) \le TC_{GI}(k_i(B)+1, B) \qquad (a) \tag{2}$$

$$TC_{GI}(k_i(B), B) \le TC_{GI}(k_i(B)-1, B). \qquad (b)$$

Goyal (1988) and van Eijs (1993) presented examples that showed that the conditions in (2) are not sufficient conditions for securing a global optimal solution for the JRP. Meanwhile, van Eijs (1993) derived another algorithm that improves efficiency of Goyal's (1974) algorithm. Later, Viswanathan (1996) derived some theoretical results that shorten the range between the upper and lower bounds on B. Viswanathan also presented an efficient algorithm that usually secures a "reasonable good" solution with a very short run time. Later, Wildeman et al. (1997) suggested a global optimization search algorithm for solving the JRP. On the other hand, Lee and Yao (2003) proposed another search algorithm that could obtain a global optimal solution for the JRP under Power-of-Two policy.

However, when solving the JRP, all the solution approaches in the literature did not take into accounts the warehouse-space restrictions.

2.2 The lot scheduling problems with the warehouse-space restrictions

One may dichotomize the relative research works on the lot scheduling problems with warehouse-space

restrictions into two categories. The first category assumes that the warehouse could replenish the products at any time point. One may refer to Zoller (1977), Geng (1984), Geng and Vickson (1988), Hall (1988) and Haji and Mansuri (1995) for the studies in this category. On the other hand, it is assumed that the replenishment in lots arrives at the warehouse only at the beginning of some basic period in the second category. Note that this study also belongs to the second category. In Section 3, we will follow this assumption to formulate our mathematical model for the JRP with warehouse-space restrictions. We have further review on the literature in the second category next.

Many researchers have been devoted to the studies on the scheduling of cyclic jobs or lots to minimize the peak requirement of production resources. For instance, one may refer to the study of the "peak load minimization problem" in Yao (2001) and the feasibility testing of the Economic Lot Scheduling Problem in Yao, Elmaghraby and Chen (2003). Park and Yun (1985) showed the problem is NP-hard when scheduling periodic, but independent activities in a single resource constrained environment. The above studies assumed that each activity (or the lot replenishment) starts and finishes within the same basic period. Recently, researchers, *e.g.*, Murthy, Benton and Rubin (2003), Yao and Chu (2008) and Yao, Chu and Lin (2008), devoted themselves to solve the lot scheduling problem so as to minimize the maximum warehouse space requirement given fixed replenishment lot sizes (and/or replenishment cycles) of multiple products. Murthy, Benton and Rubin (2003) commented that the scheduling of an activity in a given basic period does not affect the capacity requirement in basic periods in which this activity is not scheduled. It is unlike the situation in inventory replenishment of multiple products where inventory left over at the end of the last basic period becomes the inventory at the beginning of the next basic period. (In fact, such a unique characteristic makes the generation of a feasible replenishment schedule a complicated problem. We will have further discussions on this issue in Section 4.2.) Using the assumption that the replenishment cycle time of each product is an integer multiple of a basic period, Murthy, Benton and Rubin (2003) formulated a mathematical model and proposed a heuristic for solving the lot scheduling problem. Later, Yao, Chu and Lin (2008) used their experimental results to demonstrate that Murthy, Benton and Rubin's (2003) heuristic may obtain incorrect values of the maximum warehouse space requirement and there exists significant room for improving the solution quality of their replenishment schedule.

Recently, Lee and Wang (2008) present an integrated model that makes joint economic lot size decisions of the manufacturer's production batch and the replenishment lot subject to consignee's warehouse space capacity constraint so as to minimize the manufacturer's total cost. Under the assumption that the firm is allowed to adjust the selling price continuously, Transchel and Minner (2009) formulated mathematical models for the different decision frameworks to maximize the average profit by choosing the optimal pricing strategy, the optimal lot-sizes, and the optimal staggering of the order releases. They also provide algorithms to determine the optimal policy parameters, and show that the peak storage requirement is equal at each replenishment. Furthermore, they show in a numerical example that achieving operational efficiency through dynamic pricing in the warehouse scheduling problem is even more beneficial than in the economic order quantity framework. On the other hand, Hoque (2006) tried to solve the joint replenishment problem with storage and transport capacities and budget constraints. He claimed that he developed a generalized global optimal solution algorithm for the extended model and illustrated this with a numerical example. However, he was not able to prove his proposed algorithm guarantee solving an optimal solution.

A decision maker may first ignore the warehouse-space restrictions when facing the problem concerned in this study. In such a case, the decision maker could apply the solution approaches for the conventional JRP to solve the problem. However, he/she often finds that the obtained solutions are infeasible for the practice of the distribution center since the maximum warehouse-space required exceeds the available size. Therefore, we are motivated to study the JRP considering the warehouse-space restrictions and to propose a solution approach to support the managers' decision making in the distribution center.

3. THE JRP WITH WAREHOUSE-SPACE RESTRICTIONS UNDER POWER-OF-TWO POLICY

In this section, we present the derivation of the mathematical model for the JRP with the warehouse-space restrictions under Power-of-Two policy. In the first part of this section, we introduce the Power-of-Two policy. Then, we derive the constraints for the warehouse-space restrictions in the second subsection. Finally, we summarize the mathematical model for the JRP with the warehouse-space restrictions under Power-of-Two policy in Section 3.3.

3.1 Power-of-Two policy

Jackson, Maxwell and Muckstadt (1985) introduced a so-called *Power-of-Two* (*PoT*) *policy* to the JRP where *PoT*

policy rules that k_i must be a positive, power-of-two integer (*i.e.*, $k_i = 2^p, p \geq 0; p:$ integer .)

Remarkably, Jackson, Maxwell and Muckstadt (1985) provided some interesting theoretical discussion on the optimal solution resulted from the *PoT* policy. Let (K^*_{PoT}, B^*_{PoT}) be the optimal solution for the JRP under *PoT* policy and $TC_{PoT}(K^*_{PoT}, B^*_{PoT})$ be the optimal objective function value. A lower bound $\underline{TC}_{GI}$ for the optimal objective value for the JRP under *GI* policy can be easily derived by ignoring the integer restrictions on $(k_1, k_2, \ldots, k_n)$, *i.e.*, the EOQ formula for each product *i*. Provided that one is able to obtain $TC_{PoT}(K^*_{PoT}, B^*_{PoT})$, Jackson, Maxwell and Muckstadt's 6%-error theorem states that

$$\frac{TC_{PoT}(K^*_{PoT}, B^*_{PoT}) - \underline{TC}_{GI}}{\underline{TC}_{GI}} \cdot 100\% \leq 6\% \tag{3}$$

That is, the optimal objective function value under *PoT* policy is within a 6% error range from a lower bound on the optimal objecitve function value under *GI* policy. Therefore, the 6%-error theorem asserts that under *PoT* policy, one is able to obtain reasonably good solutions for the JRP. Note that using *PoT* policy could significantly simplify the generation of the replenishment schedules for the JRP with warehouse-space restrictions. (Please refer to Section 4 for further discussions.) Therefore, we define the scope of this study by using *PoT* policy.

3.2 The constraints for the warehouse-space restrictions

Note that the author referred to Murthy, Benton and Rubin's (2003) mathematical model for the derivation of the constraints discussed in this subsection. Given a vector of multipliers $(k_1, k_2, \ldots, k_n)$, we denote a replenishment schedule for such a vector as $\mathbf{X}(k_1, k_2, \ldots, k_n) \equiv (x_1(k_1), x_2(k_2), ..., x_n(k_n))$ where $x_i(k_i)$ is the earliest scheduled replenishment time point of product *i*. For a value of basic period *B*, $I_{it}(x_i(k_i), B)$ is the inventory level of product *i* at time *t* (as a function of $x_i(k_i)$ and *B* since the replenishment quantity of product *i* is $q_i = d_i k_i B$). Denote s_i as the warehouse space required for a unit of product *i*. Define $S_t(\mathbf{X}(k_1, k_2, \ldots, k_n))$ as the total warehouse space requirement at time *t*; then, $S_t(\mathbf{X}(k_1, k_2, \ldots, k_n), B) = \sum_{i=1}^{n} s_i I_{it}\left(x_i\left(k_i\right), B\right)$. Also, for a replenishment schedule $\mathbf{X}(k_1, k_2, \ldots, k_n)$, we define $S_{\max}(\mathbf{X}(k_1, k_2, \ldots, k_n), B) = \max_{0 \leq t < \infty} \{S_t(\mathbf{X}(k_1, k_2, \ldots, k_n), B)\}$ as its maximum warehouse space requirement. Let $\overline{W}_{\max}$ be the warehouse space available. If $S_{\max}(\mathbf{X}(k_1, k_2, \ldots, k_n), B) \leq \overline{W}_{\max}$, we have a feasible replenishment schedule for the solution $(k_1, k_2, \ldots, k_n, B)$. Since the replenishment cycle is assumed to be an integer multiple of a basic period *B*, we have $T_i = k_i B$ where k_i is a positive integer for all *i*. Let $lcm(\cdot)$ be an operator that takes the *least common multipliers* of a set of integers. We present the constraints for the warehouse-space restrictions as follows.

$$I_{it}(x_i(k_i), B) = d_i k_i B - d_i(t - x_i(k_i)),\ x_i(k_i) \leq t \leq (x_i(k_i) + k_i B), i = 1, ..., n\,. \tag{4}$$

$$I_{i(x_i(k_i)+k_i B)}(x_i(k_i), B) = I_{it}(x_i(k_i), B),\ 0 \leq t < k_i B, i = 1, ..., n\,. \tag{5}$$

$$\sum_{i=1}^{n} s_i I_{it}(x_i\left(k_i\right), B) \leq \overline{W}_{\max}, 0 \leq t < lcm(k_1, k_2, ..., k_n)B\,. \tag{6}$$

$$0 \leq x_i(k_i) < k_i B, i = 1, ..., n\,. \tag{7}$$

For each product, we use (4) to represent its inventory level as a function of $x_i(k_i)$ and *B* in the formulation above. Equations (5) indicate the periodic pattern of replenishment cycles. The inequalities in (6) define the maximum warehouse space requirement. At last, as shown in (7), the first replenishment time $x_i(k_i)$ of each product must be scheduled within its replenishment cycle time.

3.3 The mathematical model

Now, we are ready to summarize the mathematical model for the JRP with warehouse-space restrictions under *PoT* policy as follows.

$$\text{Minimize} \quad TC_{PoT}(k_1, k_2, \ldots, k_n, B) = \frac{1}{B}\left(\frac{A}{k_0} + \sum_{i=1}^{n} \frac{a_i}{k_i}\right) + \frac{B}{2}\sum_{i=1}^{n} k_i d_i h_i \tag{8}$$

$$\text{Subject to} \quad I_{it}(x_i(k_i), B) = d_i k_i B - d_i(t - x_i(k_i)),\ x_i(k_i) \le t \le (x_i(k_i) + k_i B), i = 1, \ldots, n \tag{9}$$

$$I_{i(x_i(k_i)+k_i B)}(x_i(k_i), B) = I_{it}(x_i(k_i), B),\ 0 \le t < k_i B, i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} s_i I_{it}(x_i(k_i), B) \le \overline{W}_{\max}, 0 \le t < lcm(k_1, k_2, \ldots, k_n)B \tag{11}$$

$$0 \le x_i(k_i) < k_i B, i = 1, \ldots, n \tag{12}$$

$$k_0 = 1 \text{ and } k_i = 2^{p_i},\ p \ge 0, P: \text{integer}, \forall i \tag{13}$$

4. THE PROPOSED SEARCH ALGORUTHM

In this section, we present the proposed search algorithm for solving the JRP with warehouse-space restrictions. Note that the proposed search algorithm is devised based on Lee and Yao's (2003) heuristic for solving the (unconstrained) JRP. Therefore, we divide our discussions in this section into two parts. In the first part, we review Lee and Yao's (2003) heuristic that obtains candidate solutions the JRP with warehouse-space restrictions. Then, in the second part, we propose a heuristic that tries to generate a feasible replenishment schedule for each candidate solution. In case that a candidate solution is infeasible, we employ a binary search procedure to seek for a *value of basic period* so that it could obtain a feasible replenishment schedule.

4.1 The search scheme for obtaining candidate solutions

Before introducing Lee and Yao's (2003) heuristic for solving the (unconstrained) JRP under *PoT* policy, we present a brief review on its theoretical background. (One may refer to Lee and Yao's 2003 paper for the details.)

For a product *i*, we collect its cost terms and express its cost function by

$$TC_{PoT,i}(k_i, B) = \frac{a_i}{k_i B} + \frac{h_i}{2} d_i k_i B \tag{14}$$

We denote $\underline{TC}_{PoT,i}(B)$ as the minimum cost function with respect to *B* for product *i*. We define a *junction point* for the $\underline{TC}_{PoT,i}(B)$ function as a particular value of *B* where two consecutive convex curves concatenate. Then, we may locate the junction point for the multipliers k_i and $2k_i$ of product *i* by

$$\delta_i(k_i) = \frac{1}{k_i}\sqrt{\frac{a_i}{h_i d_i}} \tag{15}$$

Therefore, the junction point $\delta_i(2^j)$ provides us the information that one should choose $k_i = 2^j$ for $B > \delta_i(2^j)$ and choose $2k_i = 2^{j+1}$, *vice versa*, to secure a lower value for the $\underline{TC}_{PoT,i}(B)$ function.

We define $\underline{TC}_{PoT}(B)$ as the minimum cost function of the (unconstrained) JRP under *PoT* policy; that is, $TC_{PoT}(B) = A/B + \sum_{i=1}^{n} \underline{TC}_{PoT,i}(B)$. Proposition 1 indicated the optimality structure of the $\underline{TC}_{PoT}(B)$ function as follows.

Proposition 1

The $\underline{TC}_{PoT}(B)$ *function is piece-wise convex with respect to B.* (*See also pp.* 1324 *in Lee and Yao's* 2003 *paper.*)

Some theoretical results on the junction points presented next provide the foundation for our search scheme proposed in this section.

Proposition 2

All the junction points of the piece-wise convex curve of the $\underline{TC}_{PoT,i}(B)$ *function for any product i will be inherited by the piece-wise*

convex curve of the $\underline{TC}_{P_0T}(B)$ *function.* (*See also pp.* 1325 *in Lee and Yao's* 2003 *paper.*)

Note that the objective function in (8) is separable with respective to the index of products. Therefore, we may first apply eq. (15) to obtain the sequence of the (sorted) junction points for each product separately. We may combine these *n* sorted sequences into one. Let $\{w_j\}$ be the (sorted) sequence of the junction points and $w_{j+1} < w_j, \forall j$. Note that we employ such a combined sorted sequence of junction points plays as the backbone of our search scheme. Also, Theorem 1, which is an immediate result of Proposition 2, indicates the rationale behind .

Theorem 1

Suppose that $K^{(L)}$ *and* $K^{(R)}$*, respectively, are the vectors of optimal multipliers for the left-side and right-side convex curves with regard to a junction point in the plot of the* $\underline{TC}_{P_0T}(B)$ *function. Then, there must exist a product i such that* $k_i^{(L)} = 2k_i^{(R)}$. (*See also pp.* 1326 *in Lee and Yao's* 2003 *paper.*)

Note that the objective function in (8) is separable with respective to the index of products. Therefore, we may first apply eq. (15) to obtain the sequence of the (sorted) junction points for each product separately, and combine these *n* sorted sequences into one.

Denote $k_i^*(B)$ as the optimal multiplier at a given value of *B* for product *i*, *i* = 1, 2, …, *n*. Let $K_{P_0T}(B)$ be the vector of optimal multipliers at *B*, *i.e.*, $K_{P_0T}(B) = \left(k_1^*(B), k_2^*(B), \ldots, k_n^*(B)\right)$. Then, an important by-product of Proposition 1 is an easier way to secure $k_i^*(B)$ in $K_{P_0T}(B)$, *viz.*,

$$k_i^*(B) = \begin{cases} 1, & B > \sqrt{a_i/(h_i d_i)} \\ 2^j, & \sqrt{a_i/(h_i d_i)}\big/2^j < B \le \sqrt{a_i/(h_i d_i)}\big/2^{j-1};\ j \ge 0 : \text{integer} \end{cases} \tag{16}$$

By utilizing our theoretical results above, we may easily obtain the vectors of optimal multipliers for each convex curves on the piece-wise convex curve of the $TC_{P_0T}(B)$ function. For each given vector of optimal multipliers $\mathbf{K} = (k_1, k_2, \ldots, k_n)$, we use (17) to determine an (unconstrained) optimal value of *B,* denoted by $\breve{B}(\mathbf{K})$, by taking the derivative of the $TC_{P_0T}(B)$ function with respect to *B* and equating it to zero.

$$\breve{B}(K) = \sqrt{2(A + \sum_{i=1}^{n} \frac{a_i}{k_i}) \Big/ \sum_{i=1}^{n} h_i d_i k_i} \ . \tag{17}$$

Importantly, each vector $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$ becomes a candidate solution for the JRP with warehouse-space restrictions in our proposed heuristic. Note that if we could generate a feasible replenishment schedule for each vector $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$, then the constraints from warehouse-space restrictions are inactive. In such a case, the obtained optimal solution will be the same as that of the JRP without warehouse-space restrictions. Therefore, it is reasonable to use the vectors $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$ from the JRP without warehouse-space restrictions as candidate solutions.

We proposed the following search scheme to obtain the candidate solutions $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$ by revising Lee and Yao's (2003) search algorithm for JRP without warehouse-space restrictions. (One should refer to Section 3 in Lee and Yao's 2003 paper for the details on the upper and lower bounds of the search range.)

1. Initialize the search by:
   (a) Locate the upper bound of the search range by $T_{\alpha}^* = \sqrt{2(A + \sum_{i=1}^{n} a_i)\Big/\sum_{i=1}^{n} h_i d_i}$. Set $l = 0$ and $w_0 = \min\left\{\delta_i(k_i) : \delta_i(k_i) > T_{\alpha}^*\right\}$. If there exists no such w_0, we set $l = 1$, $w_0 = T_{\alpha}^*$, $\mathbf{K}_1 = (1, ..., 1)$.
   (b) Obtain $K_{P_0T}\left(T_{\alpha}^*\right)$ by (16), and obtain $\pi = \arg\max_i \left\{\delta_i(k_i) < w_0\right\}$. Let $w_1 = \delta_{\pi}(k_{\pi})$, and j = 1. Set l = *1*, $\mathbf{K}_l = K_{P_0T}(T_{\alpha}^*)$, and compute $\breve{B}_l = \breve{B}(\mathbf{K}_l)$ by (17).
2. Proceed to the next junction point w_j and do the followings:
   (a) Compute $K_{P_0T}(w_j)$ by $K_{P_0T}(w_j) \equiv K_{P_0T}(w_{j-1}) \setminus \{k_{\pi}\} \cup \{2k_{\pi}\}$.
   (b) Obtain $\pi = \arg\max_i \left\{\delta_i(k_i) < w_j\right\}$, and let $w_{j+1} = \delta_{\pi}(k_{\pi})$.

(c) Set $l = l + 1$, $\mathbf{K}_l = K_{PoT}(w_j)$, and compute $\breve{B}_l = \breve{B}(\mathbf{K}_l)$ by (17).

3. Let $j = j + 1$. If $w_j < \breve{B}_1/2$, then stop; otherwise, go to step 2.

4.2 The generation of feasible replenishment schedules

Our focus in this subsection is to introduce the proposed feasibility testing procedure and the binary-search heuristic for the generation a feasible replenishment schedule for each candidate solution obtained by the search algorithm discussed in Section 4.1.

We provide an overview of the proposed heuristic for the generation a feasible replenishment schedule as follows. Recall that we first ignore the warehouse-space constraints and obtain a vector of optimal multipliers $\mathbf{K} = (k_1, k_2, \ldots, k_n)$ between two neighboring junction points using the proposed search algorithm. Then, we obtain a candidate solution $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$ by locating the corresponding local minimum $\breve{B}(\mathbf{K})$ using eq. (17) for each vector $\mathbf{K}$. Then, we employ a feasibility testing procedure, namely, *Proc FT* to test the feasibility of $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$. If there exists a feasible replenishment schedule for $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$, we record it as a candidate of the optimal solution. If no feasible replenishment schedule exists for $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$, we employ a binary-search procedure to locate a particular value of the basic period $\tilde{B}(\mathbf{K})$ that enables $\left(\mathbf{K}, \tilde{B}(\mathbf{K})\right)$ to obtain a feasible replenishment schedule with the minimal objective function value.

4.2.1 A feasibility testing procedure

The proposed feasibility testing procedure, namely, *Proc FT*, attempts to obtain a feasible replenishment schedule (so that it is able to test feasibility for the candidate solutions for the JRP with warehouse-space restrictions).

Given a vector of replenishment multipliers $\mathbf{K}$ and a value of basic period B, we denote $\mathbf{X}(\mathbf{K})$ as a candidate replenishment schedule and $L\left(\mathbf{X}(\mathbf{K}), B\right)$ as the maximum warehouse-space requirement (MWSR) when using the replenishment schedule $\mathbf{X}(\mathbf{K})$. An overview of the feasibility testing procedure is as follows. (One may refer to Appendix A for the details of *Proc FT*.) Assume that we are given a value of B and a vector of multipliers $\mathbf{K} = (k_1, k_2, \ldots, k_n)$ and $\bar{W}_{\max}$. We first use an '*Initial Schedule Procedure*', namely, '*Proc.IS*' (presented in Appendix A.1), to obtain an initial replenishment schedule $\mathbf{X}_1(\mathbf{K})$, and calculate the corresponding MWSR of $L\left(\mathbf{X}_1(\mathbf{K}), B\right)$. Determine L^* as the best MWSR obtained to date and $\mathbf{X}^*$ as its corresponding replenishment schedule. (Since $\mathbf{X}_1(\mathbf{K})$ is the first replenishment schedule then set $L^* = L\left(\mathbf{X}_1(\mathbf{K}), B\right)$ and $\mathbf{X}^* = \mathbf{X}_1(\mathbf{K})$ after *Proc.IS* is done). Obviously, when $L^* \le \bar{W}_{\max}$, *i.e.,* the MWSR in the planning horizon is no larger than the available warehouse space $\bar{W}_{\max}$, we obtain a feasible replenishment schedule. We define an indicator ϕ in *Proc FT*. If a feasible replenishment schedule is obtained in *Proc FT*, ϕ is equal to 1; otherwise, ϕ = 0. After *Proc.IS,* if one secures no feasible replenishment schedule, *i.e.,* ϕ = 0, we employ a '*Schedule Smoothing Procedure*' (presented in Appendix A.3) to improve L^* until ϕ = 1 or L^* can no longer be improved. If L^* has not been improved for Υ consecutive iterations, we stop the procedure (*Proc FT*) where the value of Υ is a threshold criterion for termination (which should be defined at the discretion of the analyst). Otherwise, randomly choose a subset of products for re-optimization, fix the replenishment schedule for the rest of the products, and return to start another run of local search.

4.2.2 A binary-search procedure

If one can employ *Proc FT* to generate a feasible replenishment schedule for a given vector of $\mathbf{K} = (k_1, k_2, \ldots, k_n)$ and its corresponding local minimum $\breve{B}(\mathbf{K})$, then it surely secures an optimal objective function value for the given vector of $\mathbf{K}$. But, if there exists no feasible replenishment schedule for $\left(\mathbf{K}, \breve{B}(\mathbf{K})\right)$, then we need to search for a value of B, denoted by $\tilde{B}(\mathbf{K})$, so as to enable $\left(\mathbf{K}, \tilde{B}(\mathbf{K})\right)$ to generate a feasible replenishment schedule with the minimal objective function value for $\mathbf{K}$.

Recall that $I_{it}(x_i(k_i), B)$ is the inventory level of product i at time t (as a function of $x_i(k_i)$ and B since the replenishment quantity of product i is $q_i = d_i k_i B$). By eq. (4), we have $I_{it}(x_i(k_i), B) = d_i k_i B - d_i(t - x_i(k_i))$, $x_i(k_i) \le t \le (x_i(k_i) + k_i B), i = 1, ..., n$. Obviously, the total warehouse-space

requirement at time *t*, i.e., $S_t(\mathbf{X}(\mathbf{K}))$, may be expressed by $S_t(\mathbf{X}(\mathbf{K}),B)=\sum_{i=1}^{n} s_i I_{it}\left(x_i\left(k_i\right),B\right)$. When generating a feasible replenishment schedule, the major concern is the maximum warehouse-space requirement $S_{\max}(\mathbf{X}(\mathbf{K}),B)=\max_{0\le t<\infty}\left\{S_t(\mathbf{X}(\mathbf{K}),B)\right\}$. Therefore, the easiest way to fix an infeasible replenishment schedule is to decrease the value of *B* (so as to reduce the maximum warehouse-space requirement).

For the given vector of $\mathbf{K}$, the objective function is convex with respect to *B* since it can be easily shown as positive definite. Recall that we use the binary-search procedure after learning that $\left(\mathbf{K},\breve{B}(\mathbf{K})\right)$ is infeasible. Therefore, for $B\in\left(0,\breve{B}(\mathbf{K})\right)$, we should search for the maximum value of *B* with a feasible replenishment schedule to obtain a solution with the minimal objective function value.

Having the search range $\left(0,\breve{B}(\mathbf{K})\right)$, we are ready to use a *binary search* (see Cormen, et al., 1993) to find the maximum value of *B* with a feasible replenishment schedule using *Proc FT*. We start the binary search by testing the vector $\mathbf{K}$ at $B=\breve{B}(\mathbf{K})/2$. If we obtain a feasible replenishment schedule there, we test the next value of *B* by setting $B=\left(\breve{B}(\mathbf{K})/2+\breve{B}(\mathbf{K})\right)/2=3\breve{B}(\mathbf{K})/4$; otherwise, we test $B=\left(0+\breve{B}(\mathbf{K})/2\right)/2=\breve{B}(\mathbf{K})/4$ for the vector $\mathbf{K}$. We continue the binary search until the specified error allowance is reached. Based on numerical experience, the binary search is efficient when the error allowance in *B* is set as 10^{-2}.

5. NUMERICAL EXPERIMENTS

In this section, we present an example to demonstrate the implementation of the proposed search algorithm. Also, by using random experiments, we will show that the proposed search algorithm is an effective solution approach for solving the JRP with warehouse-space restrictions.

We use MATLAB v6.5 for the coding of the proposed search algorithm. We employed a personal computer with an AMD 1800MHz CPU and 512M RAM for conducting our numerical experiments.

5.1 A demonstrative example

In this subsection, we present a numerical example to demonstrate the implementation of the proposed search algorithm. In this example, a distribution center (DC) replenishes the customers' demand of 10 products in a warehouse. Assume that the warehouse space available is $\overline{W}_{\max}=222,000$ units. Whenever the DC starts the replenishment operations for one product or a group of products, it incurs a major setup cost $A=$ \$6250 for setting up the facilities in the warehouse. The demand rate, the holding cost, and the setup cost for each individual product are shown in Table 1. (Most of the data were borrowed from Example 6 of Elmaghraby's 1978 paper.) The manager of the DC would like to determine the lot size of each product, but taking into accounts the available warehouse space when trying to save the major setup cost by jointly replenishing (groups of) products. Here, we will employ the proposed search algorithm for solving this JRP problem for the manager of the DC.

Table 1. The data set for the 10-product demonstrative example.

Product	1	2	3	4	5	6	7	8	9	10
d_i	33600	16800	4800	7200	14400	24000	72000	14400	13200	84000
h_i	0.095	0.0235	0.0065	0.022	0.023	0.075	0.1055	0.014	0.0625	0.2955
a_i	900	720	420	30	210	210	4500	2100	900	900

The search process is summarized as follows.

1. Initialize the search by:

   (a) We obtain $T_{\alpha}^{*}=24.7009$. Set $l=0$ and $w_0=\min\left\{\delta_i\left(k_i\right):\delta_i\left(k_i\right)>T_{\alpha}^{*}\right\}=26.1116$.

   (b) We set $l=l+1=1$ and $\mathbf{K}_1=K_{PoT}(T_{\alpha}^{*})$. Then, we obtain $K_{PoT}\left(T_{\alpha}^{*}\right)=(1,1,2,2,8,4,2,1,1,1,1)$ by (16),

$\breve{B}_1 = \breve{B}(\mathbf{K}_1) = 22.3135$ by (17) and $\pi = 8 = \arg\max_i \{\delta_i(k_i) \le T_{\alpha}^*\}$. Let $w_1 = \delta_\pi(k_\pi) = 22.1313$.

2. We proceed to the next junction point w_1 with the updated vector of optimal multipliers, *i.e.*, $K_{PoT}(w_1) = K_{PoT}(T_{\alpha}^*) \setminus \{k_\pi = k_8 = 1\} \cup \{k_8 = 2\} = (1,1,2,2,8,4,2,1,2,1,1)$.

   (a) We also obtain $\pi = 3 = \arg\max_i \{\delta_i(k_i) \le w_1\}$, and we locate the next junction point at $w_2 = \delta_3(2)$ where $w_2 = 20.9657$.

   (b) We set $\mathbf{K}_2 = K_{PoT}(w_1)$ and use (17) to get $\breve{B}(\mathbf{K}_2)$= 21.6767.

3. Then, we proceed to w_2 = 20.9657 with the updated vector of optimal multipliers, *i.e.*, $K_{PoT}(w_2) = K_{PoT}(w_1) \setminus \{k_\pi = k_3 = 2\} \cup \{k_3 = 4\} = (1,1,4,2,8,4,2,1,2,1,1)$. The algorithm continues its search and the whole process of the search is summarized in Table 2.

4. The termination condition in Theorem 2 is satisfied when the algorithm reaches $w_{11} = 11.0657 < \breve{B}_1/2$ (=11.15675).

Table 2. The search process of the search algorithms.

	Before applying *Proc FT*			After applying *Proc FT*		
l	$\breve{B}_l(\mathbf{K}_l)$	$TC_{PoT}(\mathbf{K}_l, \breve{B}_l(\mathbf{K}_l))$	L^*	$\tilde{B}_l(\mathbf{K}_l)$	$TC_{PoT}(\mathbf{K}_l, \tilde{B}_l(\mathbf{K}_l))$	L^*
1	22.3135	$22,475.23	271,890	17.8845	$23,027.59	219,979
2	21.6767	22,471.14	291,226	16.2588	23,406.91	220,469
3	21.5665	22,474.67	300.853	15.7352	23,600.67	219,978
4	20.0012	22,553.60	283,217	15.5332	23,278.21	219,950
5	19.4634	22,560.26	278,249	15.1500	23,272.02	220,887
6	15.7855	22,495.31	231,573	14.9300	22,530.24	220,366
7	15.4157	22,490.01	240,408	13.9592	22,600.87	221,951
8	14.8361	22,478.94	236,265	13.7206	22,547.65	221,038
9	14.6626	22,499.37	248,531	13.1180	22,638.92	221,169
10	14.6175	22,507.28	249,082	12.8974	22,683.90	221,319
11	11.2949	22,753.60	216,862	*n.a.*	22,753.60	*n.a.*

We eventually obtain 11 candidate solutions using the search scheme proposed in Section 4.1. We summarize the search process and the objective function values of these 11 candidate solutions in the left-half (*i.e.*, the part labeling with "Before applying *Proc FT*") of Table 2.

Recall that the warehouse space available is $\overline{W}_{\max} = 222,000$ units. One may notice that except the last candidate solution, *i.e.*, $(\mathbf{K}_{11}, \breve{B}_{11}(\mathbf{K}_{11}))$, it shows that $L^* > \overline{W}_{\max}$ for the other 10 candidate solutions after applying our feasibility testing procedure presented in Section 4.2.1. Therefore, we should employ our binary search procedure (presented in Section 4.2.2) to generate a feasible replenishment schedule for each one of these 10 candidate solutions. We may observe that after applying *Proc FT*, $L^* \le \overline{W}_{\max}$ holds for each revised solution.

For this example, it takes a run time of 5.23 seconds for the proposed search algorithm to obtain the best solution, namely, $\mathbf{K}_6 = (1,2,2,4,8,4,4,1,2,1,2)$ and $\tilde{B}_6(\mathbf{K}_6) = 14.9300$ with $TC_{PoT}(\mathbf{K}_6, \tilde{B}_6(\mathbf{K}_6)) = \$22,530.24$.

Note that the optimal solution of the JRP ***without*** warehouse-space restrictions serves a lower bound on the optimal objective function value of the JRP ***with*** warehouse-space restrictions. According to the left-half of Table 2, we obtained the optimal solution of the unconstrained JRP by $\mathbf{K}_8 = (1,1,2,2,8,4,2,1,2,1,1)$ and $\breve{B}(\mathbf{K}_8) = 21.6767$ with $TC_{PoT}(\mathbf{K}_8, \breve{B}_8(\mathbf{K}_8))$=$22,471.14.

Let TC_U be the optimal objective function values of the unconstrained JRP under *PoT* policy. Also, we let

TC_C be the objective function value of a solution for the JRP with warehouse-space restrictions. Then, we could make use of TC_U as a benchmark to define a performance index (which is a measure of Relative Error), namely, RE as shown in (18), for evaluating the solutions for the JRP with warehouse-space restrictions.

$$RE = \frac{TC_C - TC_U}{TC_U} \cdot 100\% \tag{18}$$

For this demonstrative example, the relative error (RE) of the solution obtained the proposed search algorithm is only 0.2282%. Therefore, the proposed search algorithm obtains a very good solution (which is very close to the lower bound).

5.2 Random experiments

Next, we use random experiments for further comparison. We refer to van Eij's (1993) paper for the parameter settings in our experiments. The annual demand rate di, the holding cost rate hi, the minor setup cost ai, for each product were randomly generated from uniform distributions, namely, UNIF[24,5600], UNIF[0.005,0.2], UNIF[5,360], respectively. We tested six levels of the number of products (n=10, 20, 30, 40, 50, 60) and five levels of major setup cost (A=250, 2,250, 4,250, 6,250, 8,250). For each combination of n and A, we generated 100 instances. Therefore, we had a total of 3,000 instances in our experiments.

If a decision maker does not have an opportunity to apply the proposed search algorithm, the most intuitive solution approach could be the Common Cycle (CC) approach in which all of the products share the same replenishment cycle $T_{cc}^* = \sqrt{2(A + \sum_{i=1}^{n} a_i) \Big/ \sum_{i=1}^{n} h_i d_i}$. Therefore, we solved each of 3,000 instances by the proposed search algorithm and the CC approach. Also, we collected the relative error (RE) and summarized our experimental results in Tables 3 and 4.

We note that the average RE (0.23% over the 3,000 instances) of the proposed search algorithm is much less than that of the CC approach, which is 20.33%. Therefore, we conclude that the proposed search algorithm significantly outperforms the CC approach in the aspect of solution quality. On the other hand, we observe that the values of the major order cost do not have substantial effects on solution quality.

Table 3. The experimental result for small-size problems ($n = 10, 20, 30$).

No. of products	Major setup cost	Average RE		Run time (seconds)
	A	CC	Yao	Yao
n = 10	250	24.320%	0.351%	5.93
	2,250	24.077%	0.510%	6.22
	4,250	24.079%	0.444%	6.38
	6,250	23.158%	0.525%	6.49
	8,250	24.386%	0.543%	5.94
	Average	24.004%	0.475%	6.19
n = 20	250	21.018%	0.260%	37.51
	2,250	21.603%	0.308%	40.76
	4,250	20.097%	0.263%	39.42
	6,250	19.480%	0.312%	41.02
	8,250	20.594%	0.290%	37.93
	Average	20.558%	0.287%	39.33

n = 30	250	19.704%	0.172%	128.49
	2,250	20.274%	0.185%	132.83
	4,250	20.069%	0.260%	127.21
	6,250	18.448%	0.262%	123.65
	8,250	18.538%	0.238%	126.37
	Average	19.407%	0.223%	127.71

The CC approach obtains its solution simply using a closed-form expression. The CC approach obtains its solution almost in no time. Therefore, we did not present the run time of the CC approach in Tables 3 and 4. On the other hand, the proposed search algorithm effectively solves the JRP with warehouse-space restrictions. For large-size problems (with n=60), it takes no more than 40 minutes to obtain a 'close-to-optimal' solution. We observe that the proposed search algorithm spent most of its run time in generating feasible replenishment schedules (i.e., in implementing the heuristic Proc FT). The curve-fitting result suggests that the average run time of Proc FT grows cubically in the problem size, i.e., the number of products. (The procedure for the polynomial-order testing can be referred to Chapter 9 "Polynomial Regression" in Neter, Wasserman and Kutner 1996.)

Table 4. The experimental result for large-size problems ($n = 40, 50, 60$).

No. of products	Major setup cost	Average RE		Run time (seconds)
	A	CC	Yao	Yao
n = 40	250	20.317%	0.164%	305.41
	2,250	17.813%	0.194%	301.78
	4,250	19.311%	0.170%	437.94
	6,250	20.314%	0.179%	306.38
	8,250	20.219%	0.189%	297.01
	Average	19.595%	0.179%	329.70
n = 50	250	19.299%	0.103%	629.88
	2,250	18.657%	0.145%	602.64
	4,250	20.513%	0.083%	636.48
	6,250	16.290%	0.137%	609.50
	8,250	20.061%	0.119%	701.90
	Average	18.964%	0.117%	636.08
n = 60	250	20.455%	0.064%	1,345.36
	2,250	18.491%	0.123%	1,120.90
	4,250	20.106%	0.100%	1,079.85
	6,250	21.249%	0.070%	1,261.73
	8,250	17.082%	0.225%	1,177.52
	Average	19.477%	0.116%	1,197.07

6. CONCLUDING REMARKS

This study is an extension of the Joint Replenishment Problem (JRP) that takes into accounts the warehouse-space restrictions in a distribution center. The focus of this study is to determine the lot sizes of each product under power-of-two policy to minimize the total costs per unit time and to generate a feasible replenishment schedule of multiple products without exceeding the available warehouse-space of the distribution center. To solve this problem, we not only derive a mathematical model, but also propose a search algorithm to obtain the optimal replenishment cycle time for each product. Also, we devise new heuristics, namely Proc FT and a binary search procedure, to generate a feasible replenishment schedule (without exceeding the available warehouse-space of the distribution center) for the candidate solutions obtained by the proposed search scheme. Using our numerical experiments, we demonstrate that the proposed search algorithm could effectively solve the JRP with warehouse-space restrictions with excellent solution quality. Therefore, the proposed search algorithm may serve as a helpful decision support tool for the decision makers.

For lot-sizing problems in supply chain management, it is important to take into accounts the warehouse-space restrictions. Currently, the author is working on extending the results of this study to cases with more general and practical scenarios in supply chains.

7. ACKNOWLEDGEMENTS

This research is supported by the National Science Council, Taiwan, Republic of China (grant NSC-93-2213-E-029-016). Also, the author would like to thank two anonymous referees for their valuable comments and suggestions.

REFERENCE

1. Aksoy, Y. and Erenguc, S. (1988). Multi-item inventory models with coordinated replenishments: a survey, *International Journal of Production Management*. 8, 63-73.
2. Elmaghraby, S.E. (1978). The economic lot scheduling problem (ELSP): review and extensions. *Management Science*, 24, 587-597.
3. Geng, P.C. (1984). Multi-Product Lot-Sizing Problems on a Single Machine. *Ph.D. Thesis, Department of Management Science*, University of Waterloo.
4. Geng, P.C. and Vickson, R.G. (1988). WRLSP: A single-machine, warehouse restricted lot scheduling problem. *IIE Transactions*, 20(4), 354-359.
5. Goyal, S.K. (1973). Economic packaging frequency for items jointly replenished. *Operations Research*, 21, 644-647.
6. Goyal S.K. (1974). Determination of optimum packaging frequency of items jointly replenished. *Management Science*, 21(4), 436-443.
7. Goyal, S.K. (1988),. Determining the optimal production-packaging policy for jointly replenished items. *Engineering Costs and Production Economics*, 15, 339-341.
8. Goyal, S.K. and Satir, A.T. (1989). Joint replenishment inventory control: deterministic and stochastic models. *European Journal of Operational Research*, 38, 2-13.
9. Graham, R.L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17, 416-429.
10. Haji, R. And Mansuri, M. (1995). Optimal common cycle for scheduling a single-machine multiproduct system with a budgetary constraint. *Production Planning Control*, 6, 151-156.
11. Hall, N., (1988). A multi-item EOQ model with inventory cycle balancing. *Naval Research Logistics*, 35, 319-325.
12. Hoque, M.A. (2006). An optimal solution technique for the joint replenishment problem with storage and transport capacities and budget constraints. *European Journal of Operational Research*, 175, 1033-1042.
13. Jackson, P., Maxwell, W. and Muckstadt, J. (1985). The joint replenishment problem with a powers-of-two restrictions. *IIE Transactions*, 17, 1, 25-32.
14. Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation; part I: graph partitioning. *Operations Research*, 37(6), 865-892.
15. Lee, W. and Wang, S.P. (2008). Managing level of consigned inventory with buyer's warehouse capacity constraint,. *Production Planning and Control*, 19, 677-685.
16. Lee, F.C. and Yao, M.J. (2003). A global optimum search algorithm for the joint replenishment problem under power-of-two policy. *Computers and Operations Research*, 30, 1319-1333.
17. Murthy, N.N., Benton, W.C. and Rubin, P.A. (2003). Offsetting inventory cycles of items sharing storage. *European*

Journal of Operational Research, 150, 304-319.
18. Neter, J., Wasserman, W. and M. Kutner (1996). *Applied Linear Statistical Models. Irvine*, Chicago, IL.
19. Park, K.S. and Yun, D.K. (1985). Optimal scheduling of periodic activities.. *Operations Research*, 33, 690-695.
20. Pinedo, M. (1993). *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, NJ.
21. Transchel, S. and Minner, S. (2009). Dynamic pricing and replenishment in the warehouse scheduling problem-A common cycle approach. *International Journal of Production Economics*, 118, 331-338.
22. van Eijs, M.J.G. (1993),. A note on the joint replenishment problem under constant demand. *Journal of the Operational Research Society*, 44, 185–191.
23. Viswanathan, S. (1996). A new optimal algorithm for the joint replenishment problem,. *Journal of the Operational Research Society*, 47, 936–944.
24. Wildeman, R.E., Frenk, J.B.G. and Dekker, R. (1997). An efficient optimal solution method for the joint replenishment problem,. *European Journal of Operational Research*, 99, 433-444.
25. Yao, M.J. (2001). The peak load minimization problem in cyclic production. *Computers and Operations Research*, 28, 1441-1460.
26. Yao, M.J. and Chu, W.M. (2008). A Genetic Algorithm for Determining Optimal Replenishment Cycles to Minimize Maximum Warehouse Space, *Omega: the International Journal of Management Sciences*, 36, 619-631.
27. Yao, M.J., Chu, W.M. and Lin, Y.F. (2008). The Feasibility Testing and Replenishment Schedule Generation in Warehouse-Space Restricted Lot Scheduling Problem. *Computers and Operations Research*, 35, 3230-3242.
28. Yao, M.J., Elmaghraby, S.E. and Chen, I.C. (2003). On the feasibility testing of the economic lot scheduling problem using the extended basic period approach. *Journal of the Chinese Institute of Industrial Engineering*, 20, 435-448.
29. Zoller, K. (1977). Deterministic multi-item inventory system with limited capacity. *Management Science*, 24, 451-455.

APPENDIX A. THE FEASIBILITY TESTING PROCEDURES

We already provide an overview of the feasibility testing procedure, namely, *Proc FT*, in Section 4.2.1. Here, we present further discussions on the components in *Proc FT*. (Recall that one implements *Proc FT* given a vector of replenishment multipliers $\mathbf{K}$ and a value of basic period *B*. Also, we denote $\mathbf{X}(\mathbf{K})$ as a candidate replenishment schedule and $L\left(\mathbf{X}(\mathbf{K}), B\right)$ as the maximum warehouse-space requirement (MWSR) when using the replenishment schedule $\mathbf{X}(\mathbf{K})$.)

A.1 The Initial Schedule Procedure (*Proc.IS*)

The "Initial Schedule Procedure" (Proc.IS) is used in two situation: (1) to obtain an initial complete replenishment schedule $\mathbf{X}_1(\mathbf{K})$ to start the local search for improvement, or (2) to obtain a "seed" replenishment schedule for the next iteration of re-improvement. It is based on a "greedy" approach that is quit fast.

Denote by $\aleph$ as the set of all n products. Let ϑ be a subset of the products, $|\vartheta| \le N$, and let $\mathbf{X}(\mathbf{K}(\vartheta))$ be a partial replenishment schedule containing only the subset ϑ of products. When Proc.IS is applied for the first time, one starts with an empty set of products $\vartheta = \varnothing$ and the empty replenishment schedule, $\mathbf{X}(\mathbf{K}(\vartheta)) = \varnothing$. Then, one assigns the warehouse requirement of products to $\mathbf{X}(\mathbf{K}(\vartheta))$ following Proc.PA (described in Appendix A.2 next), updates ϑ accordingly, until the production lots of all n products are assigned, and obtains an initial schedule $\mathbf{X}(\mathbf{K}(\aleph))$.

Next, consider the case when Proc.IS has been used at least once. Suppose that $\mathbf{X}_1\left(\mathbf{K}\right)$ is the replenishment schedule selected for re-improvement from the last iteration. To start a new iteration of improvement, one randomly selects a subset of products whose frequencies (i.e., $k_i's$) and warehouse space requirement are fixed. Let such a set be denoted by F. In the new $\mathbf{X}'\left(\mathbf{K}(\mathbf{F})\right)$, the products in the set of F are fixed at their previous replenishment schedule. Let $\bar{\mathbf{F}} = \aleph / \mathbf{F}$ be the set of products which are yet-to-be-scheduled in $\mathbf{X}'\left(\mathbf{K}(\mathbf{F})\right)$ where the symbol "/" means set subtraction. Proc.IS is now used to generate an initial schedule for the next run of re-improvement in the following manner: Let $\vartheta = \mathbf{F}$, and $\mathbf{X}'\left(\mathbf{K}(\vartheta)\right) = \mathbf{X}_1\left(\mathbf{K}(\mathbf{F})\right)$, and subtract the warehouse requirement of the products in the set F from $\aleph$. $\mathbf{X}'\left(\mathbf{K}(\vartheta)\right)$ is a new partial schedule. For the products in the set $\bar{\mathbf{F}}$, Proc.IS calls upon Proc.PA (described next) which emulates the LPT rule (that originally assigns jobs to machines) by iteratively assigning product j with the largest value of $q_i = d_i k_i B$ to the least loaded basic period

among the unassigned products in $\overline{\mathbf{F}}$. (One may refer to Graham 1969 or Pinedo 1993 for more details about the LPT rule.)

The rationale behind this heuristic rule is as follow. If a product j with a large value of q_i is scheduled after most of the products have been assigned, it may create a relative large MWSR since one may be forced to assign that large value of q_i to an already heavily loaded basic period. There is no guarantee that this heuristic rule is better than others. But, it helps in obtaining a reasonably good initial replenishment schedule.

A.2 The Product Assignment Procedure (Proc.PA)

Proc.PA constructs a (complete) schedule $\mathbf{X}(\mathbf{K})$ incrementally. Let $\mathcal{9}$ be a subset of the indices of products, $|\mathcal{9}| \le n$, and let $\mathbf{X}(\mathbf{K}(\mathcal{9}))$ be a partial replenishment schedule containing only the subset $\mathcal{9}$ of products. We define $\kappa(\mathcal{9})$ as the maximum value of the ki's in the set $\mathcal{9}$; that is, $\kappa(\mathcal{9}) = lcm\{k_i \mid i \in \mathcal{9}\}$. Suppose that product $\hat{i}$ is the next product to be added to the set $\mathcal{9}$. To assign the warehouse space requirement of product $\hat{i}$ to the replenishment schedule $\mathbf{X}(\mathbf{K}(\mathcal{9}))$, one first updates $\kappa(\mathcal{9})$ by

$$\kappa(\mathcal{9} \cup \{\hat{i}\}) = lcm\{\kappa(\mathcal{9}), k_{\hat{i}}\}. \tag{A.1}$$

If $\kappa(\mathcal{9} \cup \{\hat{i}\}) > \kappa(\mathcal{9})$, one should make $\kappa(\mathcal{9} \cup \{\hat{i}\})/\kappa(\mathcal{9})$ copies of $\mathbf{X}(\mathcal{9})$ in the entire planning horizon of $\kappa(\mathcal{9} \cup \{\hat{i}\})$ basic periods to construct a "layout" for $\mathbf{X}(\mathbf{K}(\mathcal{9} \cup \{\hat{i}\}))$. If $\kappa(\mathcal{9} \cup \{\hat{i}\}) = \kappa(\mathcal{9})$, then one employs $\mathbf{X}(\mathbf{K}(\mathcal{9}))$ directly as a layout for $\mathbf{X}(\mathbf{K}(\mathcal{9} \cup \{\hat{i}\}))$. Then, one obtains the minimal MWSR by choosing among the $k_{\hat{i}}$ ways of assignment to the layout of $\mathbf{X}(\mathbf{K}(\mathcal{9} \cup \{\hat{i}\}))$.

A.3 The Schedule Smoothing Procedure (Proc.SS)

Let $\mathbf{X}^m(\mathbf{K})$ be the schedule with minimal MWSR in a particular run of a local search (or, re-optimization) in Proc FT, and $L(\mathbf{X}^m(\mathbf{K}), B)$ be the corresponding value of MWSR given a value of B and a vector of multipliers $\mathbf{K}$. The Schedule Smoothing Procedure (Proc.SS) is used after an initial schedule $\mathbf{X}^m(\mathbf{K})$ has been obtained by Proc IS. Its aim, as the name suggests, is to "smooth" the MWSR, i.e., to minimize the maximum warehouse space requirement, among the basic periods in the planning horizon. Such "smoothing" is accomplished via two devices (subroutines) which we label the Smooth-Out Routine and the Pair-Exchange Routine. Also, in order to prevent our search being trapping in a local solution, we use a Boltzmann function when applying the Schedule Smoothing Procedure. We present these three components in the Schedule Smoothing Procedure in the following discussions.

A.3.1 The Smooth-Out Routine

The Smooth-Out Routine tries to reduce the value of MWSR by offsetting the product with the largest warehouse space requirement from the peak loaded basic period. The step-by-step procedure of the Smooth-Out Routine is presented as follows.

1. Locate the time point τ at which the MWSR occurs.
2. For those products with their k_i being larger than 1, pick the product (say, product *j*) with the largest value of warehouse space requirement at τ.
3. Examine the other (k_j–1) replenishment schedules from offsetting the replenishment time of product *j*.
   (a) If there exists at least one replenishment schedule with $L(\mathbf{X}'(\mathbf{K}), B) < L(\mathbf{X}^m(\mathbf{K}), B)$, then choose the one with the minimal value of MWSR, set $\mathbf{X}^m(\mathbf{K}) = \mathbf{X}'(\mathbf{K})$ and go back to Step 1 for the next run of improvement.
   (b) Otherwise, stop the Smooth-Out Routine.

We note that by Proposition 1, the time point τ must exist only at the beginning of some basic period in the planning horizon. Therefore, it can be easily located τ when implementing the Smooth-Out Routine. One may notice that the Smooth-Out Routine is a greedy-type local search that helps to *exploit* a neighborhood with a better replenishment schedule.

A.3.2 The Pair-Exchange Routine

By exchanging a pair of products i and j, the Pair-Exchange Routine not only assists Proc FT to reduce the MWSR, but also prevents it from trapping in a local solution.

1. Randomly pick two products *i* and *j* with both of their k_i's being larger than 1.
2. Among the other $(k_i \cdot k_j - 1)$ the replenishment schedules by offsetting the replenishment time of products *i* and *j*, pick the replenishment schedule $\mathbf{X}'(\mathbf{K})$ with the minimal value of the MWSR.
   (a) If $L(\mathbf{X}'(\mathbf{K}), B) < L(\mathbf{X}^m(\mathbf{K}), B)$, then choose the one with the minimal value of MWSR, set $\mathbf{X}^m(\mathbf{K}) = \mathbf{X}'(\mathbf{K})$ and go back to Step 1 for the next run of improvement.
   (b) Otherwise, stop the Pair-Exchange Routine.

A.3.3 A Boltzmann function

Similar to the simulated annealing approach (Johnson, et al. 1989), we use to the mechanism of Boltzmann function to prevent our search being trapped in a local solution when applying the Schedule Smoothing Procedure.

When using a Boltzmann function, we need to compute the value of β corresponding to the current solution $\mathbf{X}'(\mathbf{K})$ by comparing it with the best-on-hand solution $\mathbf{X}^m(\mathbf{K})$ in a local search as follows.

$$\beta = \exp\left\{\frac{L(\mathbf{X}^m(\mathbf{K}), B) - L(\mathbf{X}'(\mathbf{K}), B)}{\Upsilon - \chi}\right\} \tag{A.2}$$

where χ denotes the number of consecutive times that Proc FT is not able to improve $L(\mathbf{X}^m(\mathbf{K}), B)$ and Υ is the number of re-optimization allowed in Proc FT. (One will have a clear picture on the use of χ and Υ when we summarize Proc FT in Section A.4.)

Also, we need to generate a random number $p \in (0,1)$ from uniform distribution when utilizing the Boltzmann function. If $p < \beta$, we allow it to move a worse solution. Therefore, the Boltzmann function enhances the exploration ability of Proc FT. Especially, when the value of $(\Upsilon - \chi)$ is larger (which means in an earlier phase of a local search when implementing our Proc FT), we allow a large probability to change to a worse solution to explore other neighborhoods in the solution space.

A.4 A feasibility testing procedure (*Proc FT*)

Now, we are ready to summarize the proposed feasibility testing procedure, namely, Proc FT.

Before presenting the details of Proc FT, we define some new notation. Let the number of iterations of the re-optimization be indexed by γ. The value of χ denotes the number of consecutive times that the heuristic is not able to improve L*. Let $\mathbf{X}^{FT}(\mathbf{K})$ and L^{FT} be the replenishment schedule obtained by Proc FT and its MWSR, respectively.

Now, we are ready to enunciate the details of Proc FT.

Proc FT:

1. Initialization: let $\gamma = 0$, $\chi = 0$, and $\phi = 0$. Input Υ. Put all the products with $k_i = 1$ in the replenishment schedule. For the product with the maximum value of k_i, fix its first replenishment at time 0. (Select the one with largest lot size q_i additionally for tie breaking, if necessary). Try to include the other products with $k_i \geq 2$ so as to minimize the MWSR to generate a feasible replenishment schedule using the following steps.
2. Start a local search with an initial schedule $\mathbf{X}(\mathbf{K})$ obtained by *Proc.IS*.
   (a) Let $\mathbf{X}^m(\mathbf{K}) = \mathbf{X}(\mathbf{K})$ and $L(\mathbf{X}^m(\mathbf{K}), B) = L(\mathbf{X}(\mathbf{K}), B)$.
   (b) If $\gamma = 0$, then let $L^* = L(\mathbf{X}^m(\mathbf{K}), B)$ and $\mathbf{X}^*(\mathbf{K}) = \mathbf{X}^m(\mathbf{K})$.
   (c) If $L^* \leq \bar{W}_{\max}$, then let $L^{FT} = L^*$, $\mathbf{X}^{FT} = \mathbf{X}^*$, and go to Step 4.
3. Employ *Proc.SS* to improve the value of $L(\mathbf{X}^m(\mathbf{K}), B)$ corresponding to $\mathbf{X}^m(\mathbf{K})$.
   (a) Use the Smooth-Out Routine for improvement.
      i. If $L(\mathbf{X}^m(\mathbf{K}), B) \leq \bar{W}_{\max}$, then let $\phi = 1$, set $\mathbf{X}^{FT} = \mathbf{X}^m(\mathbf{K})$ and $L^{FT} = L(\mathbf{X}^m(\mathbf{K}), B)$. Go to Step 4.
      ii. Else if $L(\mathbf{X}^m(\mathbf{K}), B) > \bar{W}_{\max}$ and $L(\mathbf{X}^m(\mathbf{K}), B)$ is improved, go to Step 3.

iii. Otherwise, generate a random number $p \in (0,1)$ from uniform distribution, compute $\beta = \exp\left\{\left(L\left(\mathbf{X}^{m}(\mathbf{K}), B\right) - L\left(\mathbf{X}^{'}(\mathbf{K}), B\right)\right) \Big/ \left(\Upsilon - \chi\right)\right\}$, and check: If $p < \beta$, set $\mathbf{X}^{m}(\mathbf{K}) = \mathbf{X}^{'}(\mathbf{K})$ and update $L(\mathbf{X}^{m}(\mathbf{K}), B)$ accordingly; otherwise, go to Step (3.*b*).

(b) Use Pair-Exchange Routine for improvement.

i. If $L(\mathbf{X}^{m}(\mathbf{K}), B) \le \bar{W}_{\max}$, then let $\phi = 1$, set $\mathbf{X}^{FT} = \mathbf{X}^{m}(\mathbf{K})$ and $L^{FT} = L(\mathbf{X}^{m}(\mathbf{K}), B)$. Go to Step 4.

ii. Else if $L(\mathbf{X}^{m}(\mathbf{K}), B) > \bar{W}_{\max}$ and $L(\mathbf{X}^{m}(\mathbf{K}), B)$ is improved, go to Step 3; otherwise, go to Step (3.c).

(c) Check the improvement in L*:

i. If $L(\mathbf{X}^{m}(\mathbf{K}), B) < L^{*}$, then let $L^{*} = L(\mathbf{X}^{m}(\mathbf{K}), B)$ and $\mathbf{X}^{*} = \mathbf{X}^{m}(\mathbf{K})$. Put $\chi = 0$. Go to Step.

ii. If $L(\mathbf{X}^{m}(\mathbf{K}), B) \ge L^{*}$, put $\chi = \chi + 1$. Go to Step 4.

4. Check the termination condition:

(a) If $\phi = 1$, stop Proc FT, and output L^{FT} and $\mathbf{X}^{FT}$.

(b) If $\chi > \Upsilon$ and $\phi = 0$, stop Proc FT, and output the message: "No feasible replenishment schedule is obtained."

(c) If $\chi \le \Upsilon$ and $\phi = 0$, randomly select $\lfloor \frac{n}{2} \rfloor$ products in $\mathbf{X}^{m}(\mathbf{K})$ and let $\overline{F}$ be the subset of 'free' products. Set $\gamma = 1$. Go to Step 2 for re-improvement.